MULTIMEDIA UNIVERSITY

STUDENT ID NO

| | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | | | |

# MULTIMEDIA UNIVERSITY

# FINAL EXAMINATION

**TRIMESTER 2, 2022/2023 SESSION (T2220)**

## ECE3206 – OBJECT ORIENTED PROGRAMMING WITH C++

11 July 2023
9:00 AM – 11:00 AM
2 hours

---

**INSTRUCTIONS TO STUDENT**

1. This question paper consists of six (6) pages only (including this page).

2. **This is an open-book exam.** Hence, the questions are more in-depth than normal exams.

3. There are **THREE (3) QUESTIONS** in this paper. Answer **ALL QUESTIONS**. Question 1 carries 50 marks, Question 2 carries 30 marks and Question 3 carries 20 marks.

4. All programming questions must be answered using the C++ language. Answer all questions.

5. State all assumptions clearly.

## Question 1 (50 Marks)

a) Implement the class, member variables and member functions as specified by the following requirements. The **`main()`** function in Figure 1 is used to test your class and member functions. The expected output is as shown in Figure 2(O).

```
int main() {
Fraction f1(3, 4);
Fraction f2(1, 2);
Fraction result;

cout << "f1: ";
f1.display();

cout << "f2: ";
f2.display();

cout << "f1 + f2: ";
result = f1.add(f2);
result.display();

cout << "f1 * f2: ";
result = f1.multiply(f2);
result.display();
```

Figure 1: The `main()` function

```
f1: 3/4
f2: 1/2
f1 + f2: 10/8
f1 * f2: 3/8
```

Figure 2(O): Expected Output

i) Create a class named **`Fraction`** that represents a fraction with a numerator and denominator. The class should have two data members **`numerator`** , **`denominator`** and a constructor that initializes the data members.

[4 Marks]

ii) The class should also have member functions **`add(Fraction f)`** and **`multiply(Fraction f)`**, that return the result as **`Fraction`** object from the adding, and multiplying by another fraction respectively.

[4 + 4 Marks]

iii) Implement the **`display()`** member function that display the fraction number in the specified format as shown in Figure 2(O).

[4 Marks]

**Continued ..**

b) Implement all member functions of the **BankAccount** class as specified by the following requirements. Use the class declaration in Figure 3 as your reference. The **main()** function in b)(iv) is expected to produce the output in Figure 4(O).

```
class BankAccount {
private:
    double balance;
public:
    BankAccount()  { balance = 0.0; }
    void Deposit(double amount) ;
    void Withdraw(double amount);
    double GetBalance() ;
};
```

Figure 3: Class of BankAccount

```
Current balance: 1000
Current balance: 500
Insufficient funds.
Current balance: 500
```

Figure 4(O): Output from **main()** function

i) Implement **Deposit(double amount)** that takes a double argument representing the amount to deposit and adds it to the account balance.

[3 Marks]

ii) Implement **Withdraw(double amount)** takes a double argument representing the amount to withdraw and subtracts it from the account balance (but does not allow the balance to go negative). It will print insufficient fund if the balance is not enough.

[4 Marks]

iii) Implement **GetBalance()** that returns the account balance

[3 Marks]

iv) Implement the **main()** function that test the class and it's member functions. The **main()** function should create a **BankAccount** object and deposit RM1000 into the account. Then it withdraw RM500, followed by another withdraw of RM600. After each operation, the current balance is shown. The function should produce the output indicated in Figure 4(O).

[9 Marks]

**Continued ..**

c) Create two classes named **Point** and **Line** that represent a point and a line in two-dimensional space respectively. The **Point** class should have two data members **x** and **y** and a constructor that initializes the data members. The **Line** class should have two data members **point1** and **point2** that are objects of the **Point** class and a constructor that initializes the data members. The **Line** class should also have a member function **`length()`** that returns the length of the line. Figure 5 shows the the **`main()`** function that is used to test the classes and produces the corresponding output (as shown in Figure 6(O) ). All member functions must be implemented.

[8 Marks]

```
int main()
{
    Point p1(0, 0);
    Point p2(3, 4);

    Line line1(p1, p2);

    cout << "Length of line: " << line1.length() << endl;

    return 0;
}
```

Figure 5: The main function

```
Length of line: 5
```

Figure 6(O): Program output

d) Draw the UML diagram for the code in the question 1(c) above.

[7 Marks]

**Question 2 (30 marks)**

a) Explain the concept of inheritance in object oriented programming and give two benefits of using inheritance for efficient programming practice.

[4 Marks]

b) Create a base class called Employee with a protected variable called **`salary`**. Derive two classes called **Manager** and **Engineer** from **Employee**. Create a derived class called **Director** from **Manager**. A bonus as a multiple of the employee salary is to be paid where the engineer is given 1 month, manager 1.5 month and director 2 months salary. Add a public function called **`getBonus()`** to each class that returns the bonus of the employee. Refer to Figure 7 for the **`main()`** function that is used to test the classes and their member functions. The expected output is shown in Figure 8(O). Implement the constructors and the member functions for all the mentioned classes.

[15 Marks]

c) Draw the UML class diagram for the complete code used in question 2(b).

[8 Marks]

**Continued ..**

```
int main()
{
    Employee *employee;
    Manager manager(10000); // salary of manager $10,000
    Engineer engineer(8000); // salary of engineer $8,000
    Director director(20000); // salary of director $20,000

    employee = &manager;
    std::cout << "Manager Bonus: " << employee->getBonus() << std::endl;

    employee = &engineer;
    std::cout << "Engineer Bonus: " << employee->getBonus() << std::endl;

    employee = &director;
    std::cout << "Director Bonus: " << employee->getBonus() << std::endl;

    return 0;
}
```

Figure 7 : Code for main() function to test your classes and the expected output

```
Manager Bonus: 10000
Engineer Bonus: 8000
Director Bonus: 20000
```

Figure 8(O): Output from `main()`

d) Propose changes to the class **Employee** in question 2(b) so that it become an abstract base class.

[3 Marks]

**Continued ..**

## Question 3 (20 marks)

a) Implement the template class **Pair** that can hold a pair of two different data types. The class should have a constructor that initializes the two member variable values.

[6 Marks]

b) The Pair class in question 3(a) has two template functions in the class, namely **`getFirst()`** and **`getSecond()`**. The functions are used to retrieve the member values. Implement the template functions. Figure 8 shows the **`main()`** test function and Figure 9(O) is the corresponding output.

[6 Marks]

```
int main()
{
    Pair<int, string> p(1, "one");
    cout << "First: " << p.getFirst() << ", Second: " << p.getSecond();
    return 0;
}
```

Figure 8 : Code in **`main()`** for testing the template class and template functions.

```
First: 1, Second: one
```

Figure 9(O): Output from main()

c) Write a C++ program that calculates the division of two numbers. If the divisor is zero, throw an exception. The code given in Figure 10, which form the skeleton of your solution code must be used.

[8 Marks]

```
class Calculator
{
public:
// Define the function divide(..) that throws an exception if divide by
//zero
  double divide(double a, double b);
};

int main()
{
    Calculator calc;
   // Write try catch block  with code to test the divide() function

    return 1;
}
```

Figure 10

**End of Paper**